Name: Satyam Rai

UID: 2019130051

Subject: AI/ML

Experiment No.: 1

Aim: Implement Wumpus World problem.

Theory:

The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game **Hunt the Wumpus** by Gregory Yob in 1973.

The Wumpus world is a cave which has 4/4 rooms connected with passageways. So, there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So, the agent goal is to find the gold and climb out the cave without fallen into Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten byWumpus or falls in the pit.

PEAS Description:

1. Performance measure:

(a) +1000 points for picking up the gold — this is the goal of the agent

(b) −1000 points for dying = entering a square containing a pit or the Wumpus Monster

2. Environment: A 4 × 4 grid of squares with. . .

(a) the agent starting from square [0, 0] facing right

(b) the gold in one square

(c) the initially live Wumpus in one square, from which it never moves

(d) maybe pits in some squares. The starting square has no Wumpus, no pit, and no gold – so the agent neither dies nor succeeds straight away.

3. Actuators: The agent can turn 90° left or right walk one square forward in the current direction, grab an object in this square.

4. Sensors: The agent has 5 true/false sensors which report a stench when the Wumpus is in an adjacent square — directly, not diagonally breeze when an adjacent square has a pit glitter, when the agent perceives the glitter of the gold in the current square bump, when the agent walks into an enclosing wall (and then the action had no effect) scream, when the arrow hits the Wumpus, killing it

Code:

```
import sys
#9=stench
#7=pit
#6=gold
#5=breeze
#-1=wumpus

def learnagent(world,i,j):
        '''Function for an agent to know what poisitin contains which
environment objects'''
        if (world[i][j]==9):
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
            print("You came across a stench")
            return agi,agj
            return agi,agj
        elif (world[i][j]==7):
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
            print("You came across a pit")
            return -5,-5
        elif (world[i][j]==6):
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
            print("You found gold")
            return -4,-4
        elif (world[i][j]==5):
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
```

```
            print("You feel breeze")
            return agi,agj
        elif (world[i][j]==-1):
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
            print("You met wumpus")
            return -5,-5
        else:                                                    #if
world environment was empty
            agi,agj=i,j
            print("\nNow the agent is at "+str(agi)+","+str(agj))
            return agi,agj

def checkinp(agi,agj):
    '''Function for checking input going in forward direction to get gold'''
    if(agi==0 and agj==0):
        print("\nyou can go at  "+str(agi+1)+"  "+str(agj))            #can
move upward
        print("you can go at    "+str(agi)+"    "+str(agj+1))          #can
move right
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi+1 and agvj==agj or agvi==agi and agvj==agj+1):
            return agvi,agvj
        else:
            return -5
    elif(agi==3 and agj==0):
        print("\nyou can go at  "+str(agi-1)+"  "+str(agj))            #can go
left
        print("you can go at    "+str(agi)+"    "+str(agj+1))          #can go
right
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj+1):
            return agvi,agvj
        else:
            return -5
    elif(agi==3 and agj==3):
        print("\nyou can go at  "+str(agi-1)+"  "+str(agj))            #can go
down
        print("you can go at    "+str(agi)+"    "+str(agj-1))          #can go
left
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj-1):
            return agvi,agvj
        else:
            return -5
    elif(agi==0 and agj==3):
        print("\nyou can go at  "+str(agi+1)+"  "+str(agj))            #can go
upward
        print("you can go at    "+str(agi)+"    "+str(agj-1))          #can go
left
```

```
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi+1 and agvj==agj or agvi==agi and agvj==agj-1):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==1 and agj==0 or agi==2 and agj==0 or agi==3 and agj==0):
        print("\nyou can go at  "+str(agi+1)+"  "+str(agj))             #can go
upward
        print("you can go at    "+str(agi)+"    "+str(agj+1))           #can
move right
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi+1 and agvj==agj or agvi==agi and agvj==agj+1):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==0 and agj==3 or agi==1 and agj==3 or agi==2 and agj==3 or agi==3
and agj==3):
        print("you can go at    "+str(agi+1)+"  "+str(agj))             #can go
upward
        print("you can go at    "+str(agi)+"    "+str(agj-1))           #can go
left
        agvi=int(input("Enter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi+1 and agvj==agj or agvi==agi and agvj==agj-1):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==3 and agj==1 or agi==3 and agj==2 or agi==3 and agj==3):
        print("\nyou can go at  "+str(agi)+"    "+str(agj+1))       #can go
right
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        print("you can go at    "+str(agi-1)+"  "+str(agj))         #can move
downward
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi and agvj==agj+1 or agvi==agi and agvj==agj-1 or agvi==agi-1
and agvj==agj):
            return agvi,agvj
        else:
            return -5,-5
    else:
        print("\nyou can go at  "+str(agi)+"    "+str(agj+1))       #can go
right
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        print("you can go at    "+str(agi+1)+"  "+str(agj))         #can move
upward
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi and agvj==agj+1 or agvi==agi and agvj==agj-1 or agvi==agi+1
and agvj==agj):
            return agvi,agvj
```

```
        else:
            return -5,-5

def checkinpreverse(agi,agj):
    '''Function for checking input going in reverse direction to get back to
original position'''
    if(agi==0 and agj==3):
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi and agvj==agj-1):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==0 and agj==2 or agi==0 and agj==1):
        print("you can go at    "+str(agi)+"    "+str(agj+1))       #can go
right
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi and agvj==agj-1 or agvi==agi and agvj==agj+1 ):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==1 and agj==0 or agi==2 and agj==0):
        print("\nyou can go at  "+str(agi-1)+"  "+str(agj))         #can go
downward
        print("you can go at    "+str(agi)+"    "+str(agj+1))       #can move
right
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj+1):
            return agvi,agvj
        else:
            return -5,-5
    elif(agi==1 and agj==3 or agi==2 and agj==3):
        print("you can go at    "+str(agi-1)+"  "+str(agj))         #can go
downward
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        agvi=int(input("Enter input for row => "))
        agvj=int(input("Enter input for column => "))
        if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj-1):
            return agvi,agvj
        else:
            return -5,-5
    else:
        print("\nyou can go at  "+str(agi-1)+"  "+str(agj))         #can go
downward
        print("you can go at    "+str(agi)+"    "+str(agj-1))       #can go left
        print("you can go at    "+str(agi)+"    "+str(agj+1))       #can go
right
        agvi=int(input("\nEnter input for row => "))
        agvj=int(input("Enter input for column => "))
```

```
        if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj-1 or agvi==agi
and agvj==agj+1):
            return agvi,agvj
        else:
            return -5,-5

if __name__=='__main__':
    world=[ [0,5,7,5],
            [9,0,0,0],
            [-1,6,7,0],
            [9,0,0,7]   ]           #declaration of a world

    agi,agj=0,0                     #initial agent position
    print("\n\n\ninitially agent is at "+str(agi)+","+str(agj))
    print("\nyou can go at  "+str(agi+1)+"  "+str(agj))
    print("you can go at    "+str(agi)+"    "+str(agj+1))

    agvi=int(input("Enter input for row => "))
    agvj=int(input("Enter input for column => "))       #taking row and column
values
    if(agvi==1 and agvj==0 or agvi==0 and agvj==1):
        agi,agj=learnagent(world,agvi,agvj)             #if input valid calling
learn agent function
    else:
        print("Not valid")

    while(agi>=0):
        agvi,agvj=checkinp(agi,agj)
        if(agvi!=-5 and agvj!=-5):
            agi,agj=learnagent(world,agvi,agvj)
        else:
            print("\nNot valid")

    if(agi==-5):
        print("\nGame over Sorry try next time!!!")
    else:
        print("\nYou have unlocked next level move back to your initial
position")  #acquired gold

        agi,agj=2,1
    #implementation of reverse logic

        while(agi>=0):
            agvi,agvj=checkinpreverse(agi,agj)
            if(agvi==0 and agvj==0):
                agi,agj=-4,-4
            elif(agvi!=-5 and agvj!=-5):
                agi,agj=learnagent(world,agvi,agvj)
            else:
                print("\nNot valid")

        if(agi==-5):
```

```
            print("\nYou were really close but unfortunately you failed!!! Try
next time")
        else:
            print("\nHurray You won!!!!! Three cheers.")
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\vinay\Desktop\AIML> python -u "c:\Users\vinay\Desktop\AIML\Exp 1\wumpus_1.py"



initially agent is at 0,0

you can go at   1       0
you can go at   0       1
Enter input for row => 1
Enter input for column => 0

Now the agent is at 1,0
You came across a stench

you can go at   2       0
you can go at   1       1

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at   1       2
you can go at   1       0
you can go at   2       1

Enter input for row => 2
Enter input for column => 1

Now the agent is at 2,1
You found gold

You have unlocked next level move back to your initial position
```

```
You have unlocked next level move back to your initial position

you can go at   1       1
you can go at   2       0
you can go at   2       2

Enter input for row => 2
Enter input for column => 0

Now the agent is at 2,0
You met wumpus

You were really close but unfortunately you failed!!! Try next time
PS C:\Users\vinay\Desktop\AIML>
```

Conclusion: In this experiment I learnt about the Wumpus World problem in Artificial Intelligence. I got familiar with the PEAS concept of the same and also implemented it in Python.